APLIKASI JARINGAN SYARAF TIRUAN UNTUK PERAMALAN PERMINTAAN BARANG

Wiwik Anggraeni
Jurusan Sistem Informasi,
Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember
Kampus ITS, Jl. Raya ITS, Sukolilo – Surabaya 60111, Telp. + 62 31 5939214, Fax. + 62 31 5913804
Email: wiwik@its-sby.edu

ABSTRAK

Perhitungan jumlah barang yang harus di produksi oleh suatu perusahaan menjadi suatu hal yang sangat penting. Hal ini disebabkan oleh banyaknya akibat yang harus ditanggung oleh perusahaan dalam rangka menangani barang-barang hasil produksi tersebut, baikitu akibat dari segi ekonomi mapun social. Oleh karena itu, peramalan jumlah permintaan pada beberapa periode waktu ke depan perlu dilakukan untuk menghindari jumlah produksi barang yang berlebihan.

Dalam penelitian ini, metode yang digunakan untuk peramalan adalah Jaringan Syaraf Tiruan atau biasa disebut dengan Arificial Neural Networks dengan menggunakan algoritma backpropagation. Terhadap dataset input akan dilakukan pelatihan untuk mendapatkan bobot pada masing-masing layer sehingga didapatkan suatu nilai output. Kemudian otput tersebut akan dibandingkan dengan data tes yang diberikan dan apabila error masih besar akan dilakukan pelatihan balik dan begitu seterusnya sampai didapatkan bobot yang sesuai.

Aplikasi telah diuji coba pada system sesuai dengan spesifikasi dan kebutuhan. Hasil uji coba menunjukkan bahwa algoritma dapat diimplementasikan dengan baik dan didapatkan hasil peramalan yang mendekati data aslinya.

Kata kunci : Backpropagation, Jaringan Syaraf Tiruan, Peramalan, Permintaan produksi

1. PENDAHULUAN

Pada dasarnya JST (Jaringan Syaraf Tiruan) atau dikenal dengan ANN (*Artificial Neural Network*) adalah sejumlah satuan masukan (*input*)/luaran (*output*) terkoneksi yang setiap koneksinya memiliki bobot (*weight*) tersendiri. Disiplin ANN pada awalnya dicetuskan oleh psikologis dan neurobiologis yang mencoba mengembangkan dan menguji analogi komputasional dari jaringan syaraf atau neuron. Selama fase pembelajaran, jaringan syaraf belajar dengan mengubah-ubah bobot sehingga pada akhirnya dapat memprediksi kelas target yang tepat dari sampel-sampel masukan yang diberikan.

ANN memiliki waktu pelatihan yang cukup lama. Kebutuhan atas sejumlah parameter, seperti halnya topologi jaringan, perlu ditentukan oleh manusia terlebih dahulu. Kelemahan lain dari ANN adalah interpretabilitasnya yang rendah, karena pemahaman simbolik di balik bobot pembelajaran sangatlah sulit untuk diinterpretasikan, sehingga dapat dikatakan bahwa ANN tidak bisa menarik sebuah kaidah. Namun di balik kelemahan-kelemahan tersebut, ANN memiliki kelebihan, yaitu ANN mengijinkan toleransi yang besar terhadap data derau (*noise*). Selain itu ANN kokoh dalam hal kesalahan dalam pembelajaran data, seperti halnya manusia, serta telah diterapkan di berbagai bidang aplikasi seperti halnya interpretasi pemandangan visual, pengenalan suara, pengenalan wajah, dan strategi pembelajaran kendali robot. ANN juga kokoh dalam menangani data yang kelas targetnya bernilai angka real, angka diskret, dan vektor, bahkan untuk data sensor kompleks di dunia nyata.

2. ALGORITMA BACKPROPAGATION

Algoritma pelatihan yang digunakan adalah backpropagasi (*backpropagation*). Mengapa disebut sebagai propagasi, dapat diuraikan sebagai berikut : ketika jaringan syaraf tiruan diberikan pola masukan sebagai pola pelatihan maka pola tersebut menuju ke neuron pada lapis tersembunyi (*hidden layer*) untuk diteruskan ke neuron lapisan keluaran (*output layer*). Kemudian neuron lapisan keluaran (*output layer*) memberikan keluaran yang disebut sebagai output dari jaringan syaraf tiruan. Saat output dari jaringan syaraf tiruan tidak sama dengan output yang diharapkan, maka output akan disebarkan mundur (*backward*) pada lapis tersembunyi (*hidden layer*) diteruskan ke unit pada lapisan masukan (*input layer*). Oleh karenanya, mekanisme pelatihan tersebut dinamakan sebagai backpropagasi (*backpropagation*) atau propagasi balik.

Tahap pelatihan ini merupakan langkah untuk melatih suatu jaringan syaraf tiruan, yaitu dengan

cara melakukan perubahan bobot (*weight*) yang menyambungkan antar lapis yang membentuk jaringan syaraf tiruan melalui masing-masing unitnya.sedangkan untuk penyelesaian masalah, akan dilakukan jika proses pelatihan tersebut telah selesai. Fase ini disebut sebagai fase *mapping* atau fase pengujian/testing.

Algoritma pelatihan backpropagasi (*backpropagation*) terdiri dari 2 tahapan, yaitu feed forward dan backproragation dari galatnya. Langkah-langkah yang digunakan adalah sebagai berikut :
Langkah 0 :
Pemberian inisialisasi bobot (weight) (diberi niali kecil secara acak)
Langkah 1 :
Ulangi langkah 2 hingga 9 sampai kondisi akhir interasi dipenuhi
Langkah 2 :
Untuk masing-masing pasangan data pelatihan (training data) lakukan langkah 3 hingga 8

Umpan maju (feedforward)

Langkah 3 :
Masing-masing unit masukan (X_i, i = 1,....n) menerima sinyal masukan X_i dan sinyal tersebut disebarkan ke neuron bagian berikutnya (neuron lapisan tersembunyi)
Langkah 4 :
Masing-masing unit dilapis tersembunyi dikalikan dengan bobot (weight) dan dijumlahkan serta ditambah dengan biasnya :

$$Z\_in_j = V_{oj} + \sum_{i=1}^{n} X_i V_{ij}$$

Kemudian dihitung sesuai dengan fungsi pengaktif yang digunakan :

$$Z_j = f(Z\_in_j)$$

Bila yang digunakan adalah fungsi sigmoid maka bentuk fungsi tersebut adalah :

$$Z_j = \frac{1}{1+\exp^{(-Z\_in_j)}}$$

Sinyal keluaran dari fungsi pengaktif tersebut dikirim unit dilapis keluaran (*output layer*)
Langkah 5 :
Masing-masing unit dikeluaran (Y_k, k=1,2,3...m) dikalikan dengan bobot (weight) dan dijumlahkan serta ditambahkan dengan biasnya :

$$Y\_in_k = W_{0k} + \sum_{j=1}^{p} Z_j W_{jk}$$

Kemudian dihitung kembali sesuai dengan fungsi pengaktif

$$Y_k = f(Y\_in_k)$$

Backpropagasi (backpropagation) dan Galatnya

Langkah 6 :
Masing-masing unit keluaran (Y_k, k=1,2,3,...,m) menerima pola target sesuai dengan pola masukan saat pelatihan/training dan dihitung galatnya :

$\delta_k = (t_k - y_k)\, f'(y\_in_k)$

karena $f'(y\_in_k) = y_k$ menggunakan fungsi sigmoid, maka :

$f'(y\_in_k) = f(y\_in_k)(1- f'(y\_in_k))$
$= y_k\,(1-y_k)$

Menghitung perbaikan bobot (weight) (kemudian untuk memperbaiki w_{jk})
$\Delta W_{kj} = \alpha\,.\,\delta_k\,.\,Z_j$

Menghitung perbaikan koreksi :

$$\Delta W_{0k} = \alpha\,.\,\delta_k$$

Dan menggunakan nilai delta (d_k) pada semua unit lapis sebelumnya.

Langkah 7 :
Masing-masing bobot (weight) yang menghubungkan neuron lapis keluaran dengan neuron pada lapis tersembunyi (Z_j, j=1,...,p) dikalikan delta (δ_k) dan deijumlahkan sebagai masukan keneuron lapis berikutnya.

$$\delta\_in_j = \sum_{k=1}^{m} \delta_k W_{jk}$$

Selanjutnya dikalikan dengan turunan dari fungsi pengaktifnya untuk menghitung galat.

$\delta_j\, \delta\_in_j\, f'(y\_in_k)$

Langkah berikutnya menghitung perbaikan bobot (weight)an (digunakan untuk memperbaikiV_{ij}).

$$\Delta V_{ij} = \alpha\, \delta_j X_i$$

Kemudian menghitung perbaikan bias (digunakan untuk memperbaiki V_{0j})

$$\Delta V_{0j} = \alpha \delta_j$$

Memperbaiki *bobot (weight)* dan bias

Langkah 8 :
Masing-masing keluaran unit (Y_k, k=1,...,m) diperbaiki bias dan bobotnya (j=0,....,p),

$$W_{jk} \text{ (baru)} = W_{jk} \text{ (lama)} + \Delta W_{jk}$$

Masing-masing unit tersembunyi (Zj,j = 1,....,p) diperbaiki bias dan bobot (weight) (j = 0,....,n).

$$V_{jk} \text{ (baru)} = V_{jk} \text{ (lama)} + \Delta V_{jk}$$

Langkah 9 :
Uji kondisi pemberhentian (akhir interasi).

Daftar Notasi:

Xp	= Pola masukan pelatihan ke-p,p=1,2,...,p <=1
Xp	= (X1,X2,X3,....Xn)
tp	= Pola target dari keluaran pelatihan
tp	= (t1,t2,t3,....,tn)
Xi	= Unit ke-i pada lapis masukan
Xi	= Nilai pengaktif dari unit Xi
Zj	= unit ke-j pada lapis tersembunyi
Z_inj	= keluaran untuk unit Zj
Zj	= nilai pengaktif dari unit Zj
Yk	= unit ke-k pada lapis keluaran
Y_ink	= keluaran untuk unit Yk
Yk	= nilai pengaktif dari unitYk
Wk0	= nilai bobot (weight) pada bias untuk unit Yk
Wkj	= nilai bobot (weight) dari Zij ke unit Yk
ΔW_{kj}	= selisih antara Wkj (t) dengan Wkj(t+1)
Vj0	= nilai bobot (weight) pada bias untuk unit Zj
Vij	= nilai bobot (weight) pada unit Xi ke unit Zj
ΔV_{ij}	= selisih antara Vij (t) dengan Vij (t +1)
δ_k	= factor pengendali nilai bobot (weight)an pada lapis keluaran
δ_j	= factor pengendalian nilai bobot (weight) pada lapis tersembunyi
α	= konstanta laju pelatihan (learnig rate) 0<a<1
E	= Total galat

Metode pelatihan *backpropagation* ini disebut juga dengan *Generalized Delta Rule* (GDR). Metode yang digunakan untuk meminimalkan jumlah kuadrat kesalahan (galat) yang terjadi.

Metode *backpropagation* mencapai kondisi seperti ini dengan menghitung kemiringan permukaan galat terhadap paket-paket bobot (weight) yang tersedia pada saat itu. Selanjutnya metode ini akan dilatih dengan cara mengubah paket bobot (*weight*) mengikuti jalur tercuram (tercepat) menuju dasar lembah. Seperti telah disebutkan sebelumnya, bahwa prosesenya ada 2 tahap, yaitu :

1. Umpan maju (*feed forward*)
2. Back propagasi (*backpropagation*)

Tahap Umpan Maju

Misal H adalah himpunan pasangan pola-pola data pelatihan masukan dan targer keluaran, yang dinotasikan dengan :

$$H = \{(x1,t1),(x2,x2),(x3,x3),....,(xq,tq)\}$$

Superskrip q merupakan banyaknya pola data pelatihan.
Misalkan pola yang ke-p (p≤q) dari himpunan H adalah :

Xp = (x1,x2,x3,....,xn)
tq = (t1,t2,t3,....,tq)

Dengan xp dan tq masing-masing adalah pola data masukan dan target keluaran ke-p.
Dalam xp terdapat beberapa elemen (1,2,3,...,n) kemudian elemen tersebut menjadi masukan untuk neuron pada *input layer* yang bersesuaian. Elemen pertama menjadi masukan untuk unit pertama, sedangkan dalam tq terdapat beberapa elemen (1,2,3,....,n) target keluaran yang bersesuaian dengan neuron pada lapis tersembunyi (*hidden layer*) adalah :

$$Z\_in_j = \sum_{i=1}^{n} X_i V_{ji} + V_{j0}$$

Dengan nilai pengaktif :

$$Z_j = f(Z\_in_j)$$

$$Z_j = \frac{1}{1+\exp^{(-Z\_in_j)}}$$

Nilai pengaktif ini kemudian diteruskan ke unit pada lapis keluaran (*output layer*)
Jumlah total masukan untuk unit ke-k pada lapis keluaran (*output layer*) adalah:

$$Y\_in_k = \sum_{j=1}^{p} Z_j WK_j + Wk_0$$

Dan nilai pengaktifnya adalah:

$$Yk = f(Y\_in_k)$$

$$Z_j = \frac{1}{1+\exp^{(Y\_in_k)}}$$

Tahap Backpropagasi

Pada Lapisan Keluaran

Pada tahap ini proses awalnya adalah membandingkan keluaran JST dengan target keluaran . Galat yang diperoleh digunakan untuk memperbaiki tiap bobot (weight). Galat diminimalkan dengan persamaan :

$$E = \frac{1}{2}\sum_{k=1}^{m}(\beta_k)2$$

Dengan nilai $\beta_k = (t_k - Y_k)^2$

Persamaan berikut ini merupakan mekanisme BP dalam memperbaiki nilai tiap bobot (weight)nya. Diasumsikan untuk satu perbaikan bobot (weight) dari unit j ke unit k.

$$E = \frac{1}{2}\sum_{k=1}^{m}(t_k - y_k)^2$$

Sehingga gradien negatif E untuk tiap unit pada lapisan keluaran (*output layer*) :

$$\frac{\partial E}{\partial W_{kj}} = -\frac{\partial}{\partial W_{kj}}\left[\frac{1}{2}\sum_{k=1}^{m}(t_k - y_k)^2\right]$$

$$\frac{\partial E}{\partial W_{kj}} = -\frac{\partial}{\partial W_{kj}}\left[\frac{1}{2}\sum_{k=1}^{m}(t_k - f(y\_in_k))^2\right]$$

$$\frac{\partial E}{\partial W_{kj}} = -\sum_{k=1}^{m}(t_k - y_k)^2 f'(y\_in_k)\frac{\partial}{\partial W_{kj}}(y\_in_k)$$

$$\frac{\partial E}{\partial W_{kj}} = -(t_k - y_k)f'(y\_in_k)Z_j$$

Bobot dapat diperbaharui dengan persamaan:

$$W_{kj}(t+1) = W_{kj}(t) + \Delta W_{kj}$$

$$\Delta W_{kj} = -\alpha\frac{\partial E}{\partial W_{kj}}$$

$$\Delta W_{kj} = \alpha(t_k - Y_k)f'(Y\_in_k)Z_j$$

$$\Delta W_{kj} = \alpha.\partial_k.Z_j$$

Sehingga

$$W_{kj}(t+1) = W_{kj}(t) + \alpha\partial_k Z_j$$

Dengan alpha (learning rate) adalah suatu parameter yang digunakan untuk mengatur laju perubahan nilai bobot dan bernilai antara 0 dan 1.

Pada Lapis Tersembunyi

Bobot (weight) pada lapis tersembunyi diperbaiki dengan cara yang serupa seperti pada lapis keluaran, tetapi nilai target keluaran tidak diketahui. Nilai galat (E) dapat diketahui berkaitan denga nilai pengaktif pada lapis ini dengan persamaan minimisasi galat

$$E = \frac{1}{2}\sum_{k=1}^{m}(t_k - Y_k)^2$$

Persamaan untuk memperbaiki bobot adalah :

$$\Delta V_{ij}(t+1) = V_{ji}(t) + \alpha\delta_j X_i$$

Parameter pelatihan

Parameter-parameter yang turut menentukan keberhasilan proses pelatihan pada algoritma BP :

- Inisialisasi bobot (weight)
- Jenis adaptasi bobot (weight)
- Learning rate/laju pelatihan
- Momentum
- Penentuan jumlah lapis tersembunyi

Inisialisasi bobot

Bobot sebagai interkoneksi jaringan syaraf tiruan yang akan dilatih biasanya diinisialisasi dengan nilai nyata yang kecil dan diinisialissi secara acak.

Pada banyak penelitian (Hirose et al, 1991) menunjukkan bahwa konvergensi tidak akan dicapai bila nilai bobot kurang bervariasi dan juga bila nilai acaknya terlalu kecil. Konvergensi hampir selalu tercapai untuk inisialisasi acak pada -0.5 sampai 0.5 atau -1 samapai 1.

Parameter laju pelatihan (η)

Parameter laju pelatihan (*learning rate*) sangat berpengaruh pada intensitas proses pelatihan. Begitu pula terhadap efektifitas dan kecepatan mencapai konvergensi dari pelatihan

Nilai optimum dari *learning rate* tergantung masalah yang diselesaikan, prinsipnya dipilih sedemikian rupa sehingga tercapai konvergensi yang optimal dalam proses pelatihan.

Nilai *learning rate* yang cukup kecil menjamin penurunan gradien terlaksana dengan baik, namun hal ini berakibat bertambahnya jumlah itersi. Pada umumnya besar nilai *learning rate* tersebut dipilih mulai 0.001 sampai 1 selama proses pelatihan.

Momentum

Disamping *learning rate* terdapat koefisien lain yang tujuan penggunaannya untuk mempercepat konvergensi dari algoritma error backpropagation. Prinsip dari metode ini adalah menambahkan sebagian dari perubahan bobot sebelumnya. Hal ini dapat dirumuskan dengan :

$$\Delta w(t) = \eta\nabla E(t) + \alpha\Delta w(t-1)$$

Dengan α adalah nilai konstatnta momentum yang berupa bilangan positif antara 0.5 dan 0.9

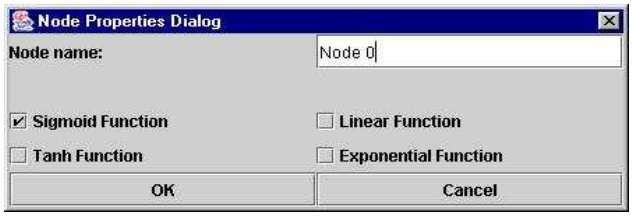
Penggunaan koefisien momentum disarankan jika konvergensi berlangsung terlalu lama, dan juga untuk mencegah terjadinya optimum lokal(local optimum/minimum)

3. Desain Topologi Jaringan pada Aplikasi

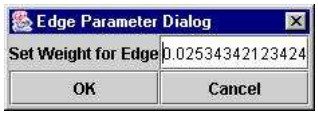
Dalam aplikasi ini, mula-mula topologi jaringan harus dibuat terlebih dahulu. Seperti yang telah dijelaskan di atas, sebuah topologi jaringan secara teknis terdiri dari node input, node hidden layer, node output, dan edge penghubung antar node.

Untuk membuat sebuah node, klik toolbar Create Node, lalu klik di kanvas sehingga muncul dialog properti node seperti di gambar 3. Untuk membuat sebuah edge, klik Create Edge, lalu klik start-point node dan klik end-point node. Otomatis start-point node berubah menjadi bentuk layang-layang dan end-point menjadi sebuah persegi besar. Untuk mengubah properti dari node ataupun edge, klik kanan entitas tersebut pilih Set Property of Entity (lihat gambar 2 dan gambar 3).

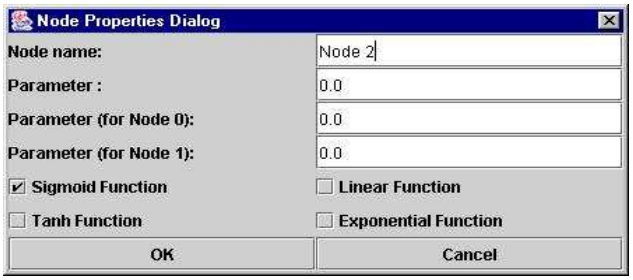
Pada kesempatan kali ini penulis akan melakukan pengujian dengan menggunakan beberapa design, seperti terlihat pada gambar 3,4, ean 5. Klik tab Solve untuk melakukan pelatihan. Bobot dapat diacak melalui tombol Bobot Random. Opsi pembelajaran dapat diisi di menu Neural Options → Learning Options. Di sana ada nilai learning rate (η)(learning rate yang digunakan dalam pengujian adalah 0,9), momentum (α), dan batasan pengacakan bobot (plus-minus). Begitu pula, stopping condition dapat diset di menu Neural Options → Stopping Conditions. Di sana ada number of iterations dan target error(target error yang digunakan dalam pengujian adalah 0,1). Kecepatan proses juga bisa dipilih, sesuai keinginan.



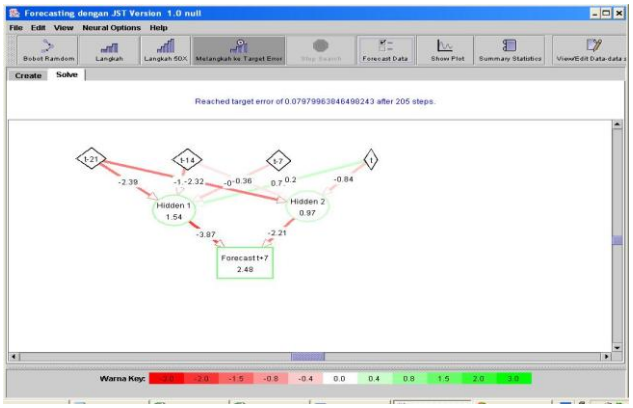
Gambar 1 : (Input) Node Properties Dialog



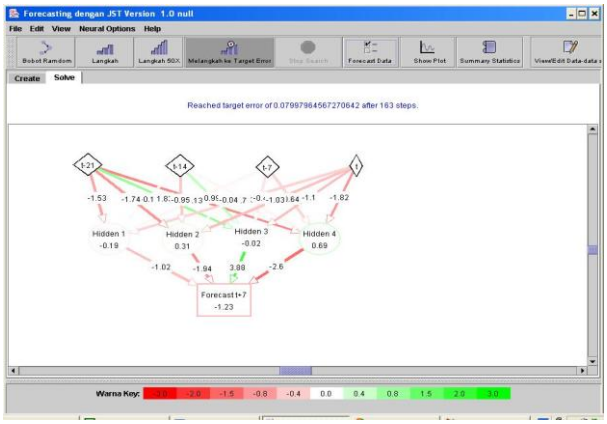
Gambar 2 : Edge Parameter Dialog



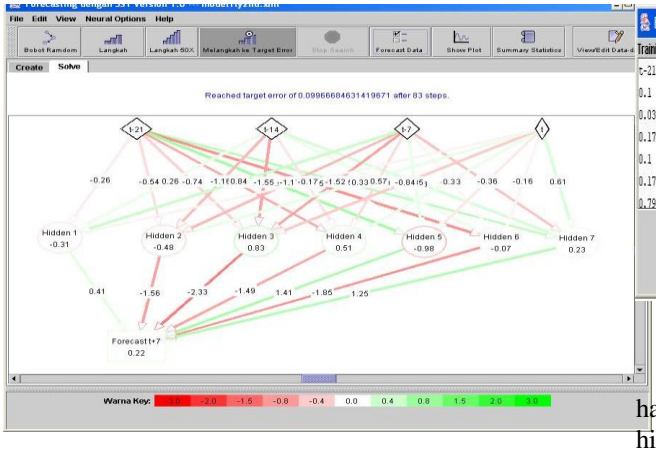
Gambar 3 : Hidden Layer Properties Dialog



Gambar 4 : Desain topologi jaringan dengan 2 node 1 hidden layer



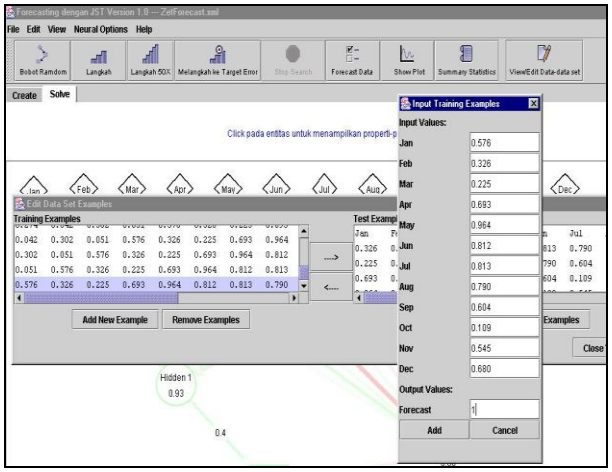
Gambar 5 : Desain topologi jaringan dengan 4 node 1 hidden layer



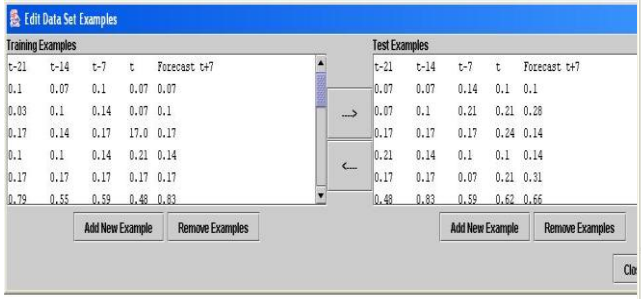
Gambar 6 : Desain topologi jaringan dengan 7 node 1 hidden layer

4. DATASET

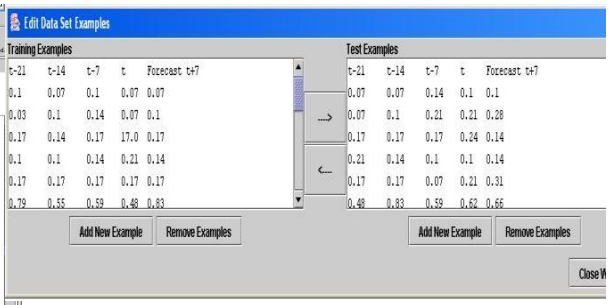
Sampel pelatihan dan sampel uji yang diberikan bisa dimasukkan melalui menu View / Edit Dataset seperti pada gambar 8 atau bisa juga diedit melalui XMLnya. Kali ini penulis akan melakukan pengujian menggunakan supervised dataset dan unsupervised dataset(dapat dilihat pada gambar 9 dan 10).



GGambar 8 : Menu untuk memasukkan dataset



Gambar 9 : Supervised Dataset



Gambar 10 : Unsepervised Dataset

Data yang dientrikan sebagai dataset haruslah sudah dinormalisasi, yaitu memiliki nilai 0 hingga 1 (seperti yang ditunjukkan pada gambar 9 dan 10). Normalisasi dapat dilakukan dengan menggunakan normalisasi min-max. Rumusnya adalah :

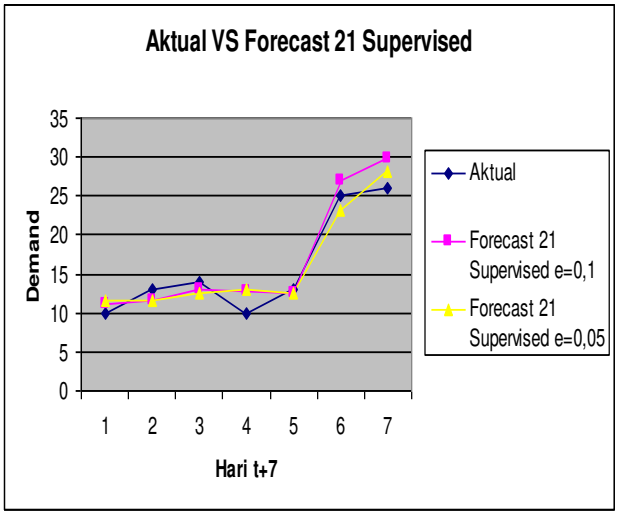
$$v' = \frac{v - \min_A}{\max_A - \min_A}(new\_\max_A - new\_\min_A) + new\_\min_A$$

5. Uji Coba

Uji coba dilakukan pada topologi jaringan dengan 2 node 1 hidden layer, 4 node 1 hidden layer, 6 node 1 hidden layer, 2 node 2 hidden layer dan 6 node 2 hidden layer dengan *stopping condition* yang berbeda-beda. Untuk topologi satu hidden layer dengan 2 unit hidden layer, kita coba melakukan iterasi hingga error mencapai 0.1. Makin beranjak ke iterasi berikutnya, total error akan makin mengecil secara konvergen.

Gambar 11: Tingkat Penurunan Galat

Pada percobaan kali ini, penulis akan melakukan uji coba peramalan dengan menggunakan target error sebesar 0,1 dan 0,05. Hasilnya diperoleh grafik Aktual VS Forecast 21 Supervised e=0.1 VS Forecast 21 Supervised e=0.05 yang ditunjukkan pada gambar 12.



Gambar 12 : Aktual VS Forecast 21 Supervised e=0.1 VS Forecast 21 Supervised e=0.05

Daftar pengukuran error yang didapatkan dapat dilihat pada tabel 1 di bawah ini.

Tabel 1 : Daftar pengukuran error untuk learning rate = 0.1 dan 0.05

Jenis Error	Error = 0.1	Error = 0.05
Mean Squared Error	4,4	3,5
Mean Percentage Error	-6%	-3%
Mean Absolute Percentage Error	12%	13%

Untuk *stopping condition* yang sama, jika hidden layer ditambah 1 layer dengan sejumlah unit node ternyata jumlah iterasi dapat mencapai dua kali lipatnya. Perbandingan error untuk *stopping condition* 0.1 dapat dilihat pada tabel 2 berikut.

Tabel 2 : Daftar pengukuran error untuk jumlah hidden layer yang berbeda

Jenis Error	1 Hidden Layer	2 Hidden Layer
Mean Squared Error	88.19	60.29
Mean Percentage Error	-11.31%	-8.64 %
Mean Absolute Percentage Error	11.65 %	9.28 %

6. KESIMPULAN

Beberapa kesimpulan yang dapat ditarik dari penelitian ini adalah sebagai berikut.:

1. Hasil pengujian dengan menggunakan supervised dataset memakan waktu lebih cepat dan error yang lebih kecil jika dibandingkan hasil pengujian menggunakan unsupervised dataset.
2. Untuk stopping condition dengan error yang kecil statistic errornya hamper sama
3. Semakin banyak jumlah node yang ada di dalam hidden layer, hasil peramalan mendekati data aslinya
4. Semakin kecil nilai rate dan momentum yang diberikan, error peramalan akan semakin kecil.

7. DAFTAR PUSTAKA

1. Bruce LB, Ricard O Connell,Anne Koehler, “Forecasting, time series, and Regressions”, Koehler Duxbury Press 2003
2. Robert A. Yaffee, “An Introduction to Time Series Analysis and Forecasting: with Applications of SAS and SPSS”, Academic Press, 2002
3. Hanke, Wichern, and Reitsch, “Business Forecasting”, 7th Ed , 2001
4. cortex.snowseed.com/neural_networks.htm: Introduction to Backpropagation Neural Networks
5. www.willamette.edu/~gorr/classes/cs449/intro.html